Design and Implementation of Neural Network Accelerator for Binocular Matching

Kaijian Zeng, Yaofeng Hou

***Abstract*— Convolutional Neural Networks (CNNs) have demonstrated significant advantages in binocular stereo matching tasks, but the high computational complexity of their 3D extensions (3D CNNs) limits real-time applicability. This study proposes a 2D convolutional cost feature-based binocular matching neural network accelerator, achieving efficient deployment through algorithm-hardware co-design. Key innovations include: (1) Designing the FDSCS network architecture with optimized cost volume generation modules, integrating enhanced preprocessing and pipeline mechanisms; (2) Restructuring 2D convolution via an Img2Col-GEMM strategy to leverage FPGA parallel computing for accelerated matrix operations; (3) Introducing network weight quantization and bilinear interpolation modules to reduce memory requirements while improving output accuracy. Evaluations on the ZCU102 platform demonstrate that the accelerator achieves 18.72 ms per-frame processing speed and a 3.22% average error rate on the KITTI dataset, balancing real-time performance with precision for time-sensitive applications such as autonomous driving.**

***Index Terms*—Binocular Matching, CNN, FPGA.**

I. Introduction

Binocular stereo matching is a fundamental technique in computer vision, with its core objective being to calculate the parallax between corresponding pixel points in images captured from two different viewpoints by computing and aggregating matching costs, thereby recovering the 3D depth information of the scene. This process entails complex image processing and matching algorithms, with early methods primarily relying on local feature matching[1] and global optimization strategies[2]. However, traditional methods often struggle to maintain matching accuracy and stability in complex scenarios, such as occlusion, repeated textures, and lighting variations.

With the advent of deep learning techniques, particularly the rapid advancement of convolutional neural networks (CNNs), deep learning-based binocular matching methods have become increasingly prevalent[3]. These methods significantly improve the robustness and accuracy of matching by learning rich image feature representations. Early deep learning models, such as DispNet[4], directly predicted pixel-level depth maps, while later models, such as GC-Net[5], introduced the cost volume concept, processed using 3D convolutional networks to achieve more precise parallax estimation. However, the application of these models in real-time systems is constrained by their high computational complexity and stringent hardware resource demands

Considering the low accuracy of traditional binocular matching algorithms in low-texture and occluded regions, as well as the limitations of 3D convolutional networks in real-time applications, this study proposes the design of an efficient binocular matching neural network accelerator based on 2D convolutional cost features, achieving an optimal balance between accuracy and real-time performance. The main contributions of this study can be summarised as follows:

(1) Algorithm-hardware co-optimization: We conduct a detailed analysis of the FDSCS network architecture and design a dedicated hardware implementation strategy for its key operators, significantly reducing computational latency and resource consumption through customized hardware modules and optimized memory access patterns.

(2) High-efficiency convolutional implementation: We employ GEMM as the basis for convolution operations, combining it with techniques such as loop unrolling and data caching to fully leverage the parallel processing capabilities of FPGAs, thereby enhancing processing efficiency.

(3) Comprehensive performance evaluation: We evaluated our binocular matching method on the ZCU102 FPGA platform to demonstrate its effectiveness and efficiency. The experimental results indicate that our method achieves a processing speed of 53.42 frames per second (fps) for 1242 × 375 resolution images over a range of 128 parallaxes, meeting the real-time processing requirements of most systems. Additionally, the average error rate is 3.22%, which compares favorably to other implementations.

II. Related Work

To address the computational and deployment challenges of deep learning models, research on hardware accelerators has garnered significant attention. Hardware accelerators, particularly those based on field-programmable gate arrays (FPGAs) and ASICs, offer superior energy-efficiency and computational performance compared to general-purpose processors (CPUs or GPUs) due to their customized designs. In the field of deep learning, FPGAs are an ideal platform for implementing efficient neural network accelerators due to their flexibility and reconfigurability.

In recent years, significant progress has been made in accelerator designs for binocular stereo matching, encompassing diverse methodologies. For instance, Cambuim et al.[6] developed an occlusion-robust stereo vision system that enhanced robustness and processing speed, achieving 25 FPS at 1024×768 resolution and 159 FPS at

Manuscript received March 09, 2025

Kaijian Zeng, School of computer science and technology, Tiangong University, Tianjin, China.
Yaofeng Hou, School of computer science and technology, Tiangong University, Tianjin, China.

320×240 resolution. While this approach demonstrated advantages in resource efficiency, its performance degraded in low-texture and occluded regions due to insufficient global context. In contrast, Liang et al.[7] proposed an adaptive window mechanism integrated with left-right consistency checks and median filtering post-processing, improving average matching accuracy by 5.07% compared to conventional algorithms. Their implementation on a Zynq UltraScale+ chip achieved 54.24 frames per second (FPS) for 1280×720 resolution images with 64 disparity levels. Meanwhile, Rahnama et al.[8] implemented a semi-global matching (SGM) algorithm seeking to balance local and global optimization for enhanced accuracy. However, all these FPGA-based accelerators rely on handcrafted features and traditional algorithms, which fundamentally limits further precision improvements.

III. ARCHITECTURE

A. Overall Algorithm Flow

As depicted in Fig.1, our stereo matching framework comprises cost volume construction based on a local matching method, cost feature transformation and aggregation using a CNN engine, and post-processing with bilinear interpolation. In this chapter, we will describe the implementation of each module in detail.

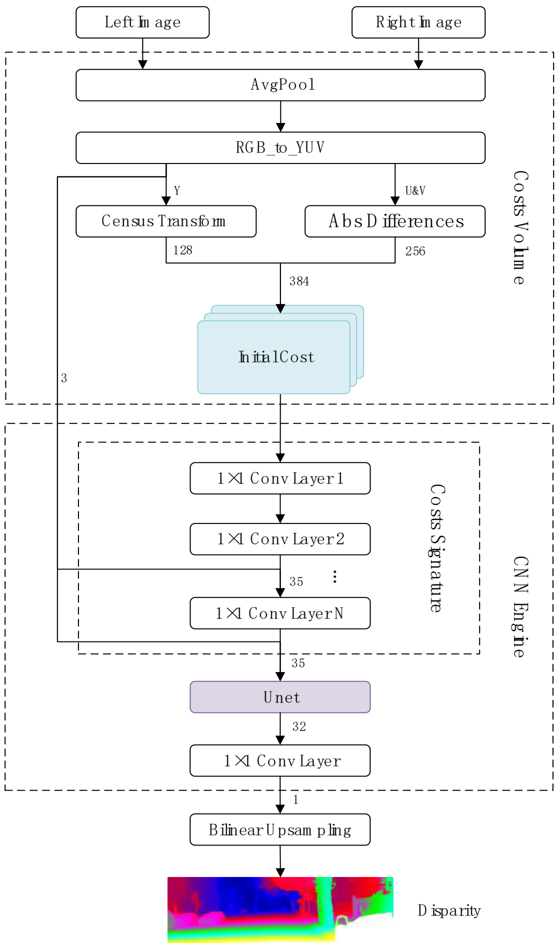

Fig.1 Overall Algorithmic Flow

B. Image Pre-processing Module Design

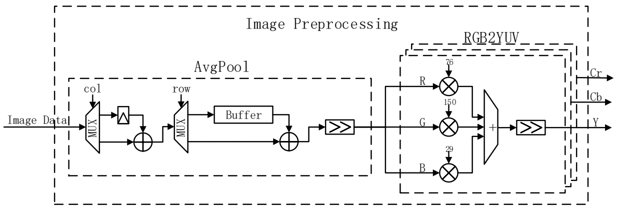

Fig.2 Implementation of the image pre-processing module

The design architecture of the image preprocessing module is shown in Fig.2. First, the size of the input left and right images is reduced using the average pooling (AvgPool) technique. A sliding window is defined, and the average of all pixel values within each window region is computed. This average represents the new pixel value for that region. In this way, the original image is scaled to a smaller size, but the overall structure and texture information is still preserved. During this process, a temporary buffer is used to construct the sliding window, and the image data is imported into the buffer via a row and column selector. Next, the pixel values are accumulated using an adder and the average value is calculated by a right-shift operation.

Next, the image is converted from RGB color space to YCbCr color space. This conversion is particularly important for stereo vision tasks because it decomposes the image into subjective luminance (Y) and color difference (Cb, Cr) components, helping to separate the effects of luminance variations in the subsequent Census Transform. Notably, to adapt to the fixed-point arithmetic characteristics of the hardware, an approximate color conversion formula is employed to simplify the originally complex floating-point operation into a fixed-point one, improving both computational efficiency and hardware implementability.

C. Census Transform Module Design

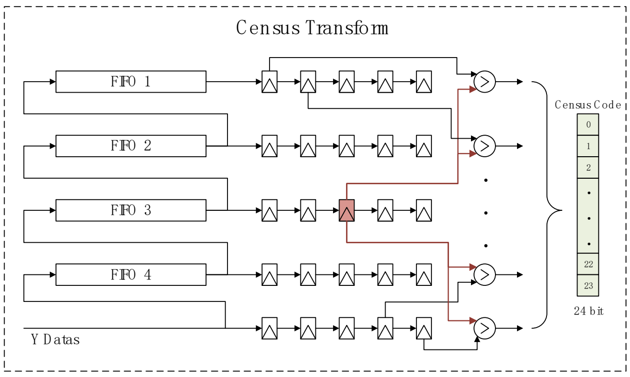

Fig.3 Implementation of the Census Transform module

Fig.3 illustrates the architecture of the Census transform, designed to efficiently generate Census codes for stereo matching. The design employs four FIFO buffers to temporarily store four rows of pixel data. Once these rows are cached, the system simultaneously outputs five adjacent row data columns aligned vertically. These data are then fed into a 5×5 register array to form the processing window. At this stage, the center pixel is compared with all other pixels within the window to generate the Census code. Specifically, for each sliding window centered on a target pixel, the comparison results between the center pixel and its neighbors (assigned a binary "0" if the neighboring pixel's grayscale value is less than or equal to the center pixel, otherwise "1") are concatenated bitwise into a binary string, forming the Census code. This process assigns each pixel a unique Census signature that encapsulates its local structural information.

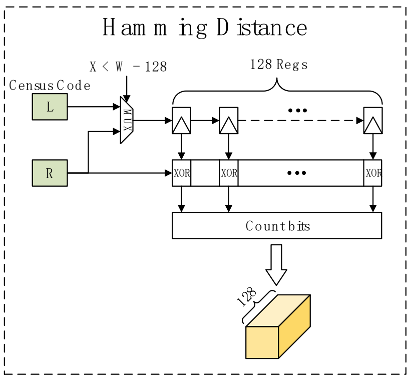

Fig.4 Implementation of the Hamming Distance module

To further process the image data, we employ 128 shift registers to buffer the Census codes and calculate the Hamming distance between left-right image Census codes at corresponding positions through bitwise XOR operations. The computational workflow is detailed in Fig.4.

Specifically, the Census codes from the left image are first cached sequentially using 128 shift registers. After buffering, the stored left-image Census codes are streamed alongside the corresponding right-image Census codes into XOR gates for bitwise comparison. Since Hamming distance quantifies the number of differing bits between two equal-length binary strings, we count the number of '1's in the XOR results to determine this metric. The Hamming distance directly reflects the similarity between matched positions: a larger Hamming distance indicates lower similarity.

To preserve original image dimensions during edge processing, the shift registers store left-image Census codes when the horizontal coordinate satisfies $X < W - 128$ (where W denotes image width). For the final 128-pixel columns ($X \geq W - 128$), right-image Census codes are stored instead, with zeros serving as padding values.

D. Design of CNN Engine

Fig.5 illustrates the internal architecture of the pulsed array and its processing elements (PEs). Each PE cell receives operands from its port, performs multiplication and addition operations, and transmits the operands to neighboring PE cells in the following cycle. By arranging the PE cells in a Height × Width *matrix*, a two-dimensional pulsed array structure is designed. Additionally, a hierarchical design concept is employed to combine these 2D arrays vertically, forming a 3D pulsating array topology with dimensions Tile × Height × Width. This topology enhances data processing parallelism and improves both throughput rate and computational density. Cache Unit A stores the output data from the img2col unit and facilitates efficient data transfer to the computational array using a ping-pong caching mechanism and a MUX select-through function. Cache Unit B stores the weights required for convolutional computation, with each cache block storing the weights of one output channel, enabling the computation of convolutional kernels for up to Tile × Width output channels per operation.

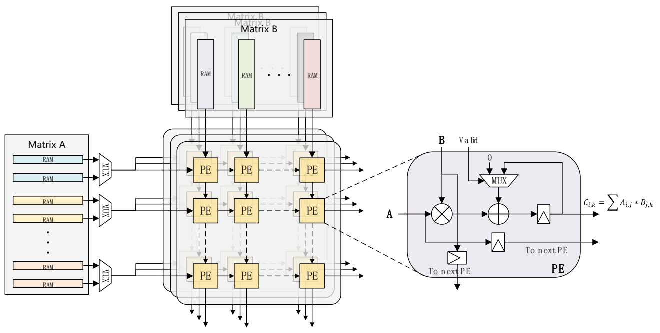

Fig.5 3D pulsed array architecture

E. Post-processing Module Design

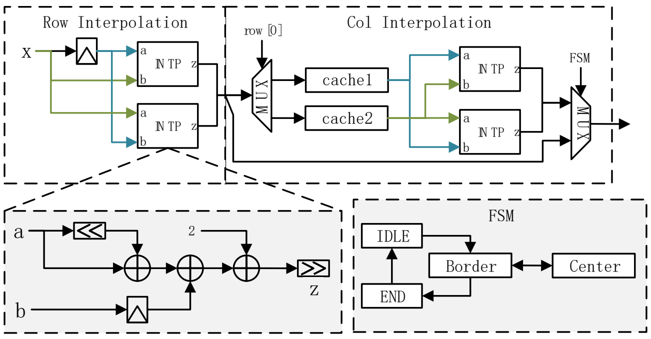

Fig.6 2×2 Bilinear interpolation architecture

Fig.6 illustrates the architecture of bilinear interpolation. Shifters and adders are employed to substitute multiplication and division operations in the interpolation formula, thereby reducing resource consumption and enabling rounding of the calculation results. By utilizing a selector and two buffers to alternately store the results of horizontal interpolation, continuous output of neighboring points in the vertical direction is achieved, thereby completing the interpolation operation vertically. The entire interpolation process is managed by a state machine, which ensures both the accuracy of the interpolation results and the correct output order.

IV. Analysis Of Experimental Results

A. Experiment Settings

This study presents a comprehensive evaluation of the hardware accelerator design, focusing on resource efficiency and performance. We used Spinal HDL as the RTL source code generation tool and Xilinx Vivado 2021.2 for synthesis and implementation. The accelerator was implemented on a ZCU102 platform, with an operating frequency of 200 MHz. The architectural parameters of the pulsed array were set to [Tile, Height, Width] = [8, 8, 32]. We evaluated the accuracy of the accelerator using the KITTI2015 dataset to compute the parallax error rate, a widely recognized benchmark in stereo vision due to its complexity and practical utility[9]."

B. Resource Utilization

Fig.7 illustrates the resource usage of the hardware accelerator, evaluated under various systolic array configurations and image resolutions. Fig.7 (a) shows the impact of different systolic array configurations on resource consumption at 1242×375 resolution. The results revealed a significant increase in frame rate from 1.48 FPS (1, 8, 8) to 53.42 FPS (8, 8, 32), a nearly 36-fold increase. This increase was primarily due to the larger systolic array size, significantly increasing the number of DSPs (arithmetic units). However, despite the larger systolic array size increasing hardware resource usage, it significantly reduced computational latency.

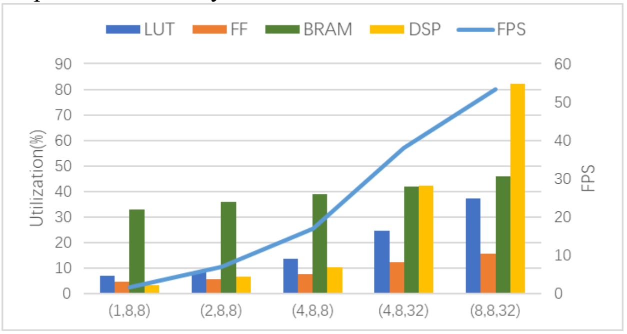

Fig.7 (a) Resource utilization under different pulsation array configurations

Fig.7 (b) shows the impact of different resolutions on resource utilization under the (8, 8, 32) configuration. The results showed that while the number of DSPs, FFs (registers), and LUTs (look-up tables) increased by approximately 10%, BRAM requirements grew disproportionately. This was attributed to using line buffers for storing feature maps, with buffer size depending on the number of channels and feature map width. Under the (8, 8, 32) configuration and 1242×375 resolution, our design consumed 37.3% of LUTs, 15.6% of FFs, 46% of BRAM, and 82.2% of DSPs, allowing the system to achieve 53.42 FPS.

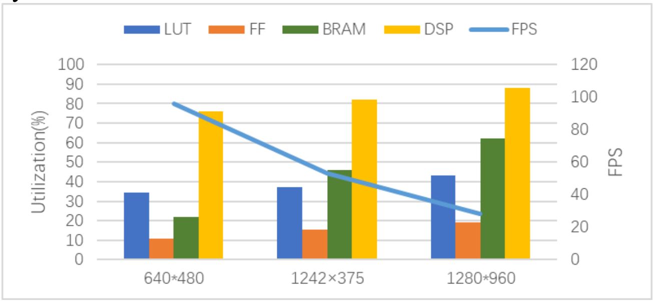

Fig.7 (b) Resource utilization at different resolutions

C. Performance Evaluation

Table 1 Comparison of parallax error rate and processing speed with other methods

Method	Error rate	Platform	Runtime(ms)
FSDCS	3.22	Xilinx ZCU102	18.72
iELAS[10]	19.8	Xilinx Virtex-7	17.39
FP-Stereo[11]	9.81	Xilinx ZCU102	6.21
Lite-Stereo[12]	6.41	Altera Stratix V	7.1
Census + wSGM[13]	6.54	Xilinx VCU-118	-
StereoEngine[14]	6.37	Altera Stratix V	6.07
FSDCS	3.16	Nvidia Titan X	23.21

To assess the performance of our proposed system, we conducted tests on various hardware platforms and compared them to existing stereo vision systems. Accuracy was measured by calculating the percentage of pixels with a disparity error exceeding three pixels across all regions. Runtime was defined as the time needed to process image pairs at a resolution of 1242 × 375. As indicated in Table 1, our CNN model achieved a 3.22% error rate on the KITTI2015 dataset, outperforming FPGA-based solutions reported in the literature ([10] - [14]). Although slower in terms of processing speed, our system significantly enhanced accuracy due to its stronger representation capabilities. When deployed on an Nvidia Titan X, despite only a slight improvement in error rate, our system demonstrated higher efficiency in power consumption and hardware resource utilization compared to GPU platforms, offering advantages in practical applications.

Table 2 Comparison of different stereo matching FPGA implementations

	Method	Disparities	Resolution	FPS	MDE/s
Ours	FSDCS	128	1242 × 375	53.42	3,185
			1280 × 960	28.25	4,443
[15]	AD-Census	64	480 × 270	41	340
[16]	SAD	64	1242 × 375	19.61	584
[7]	SGM	64	1280 × 720	54.24	3,199
[10]	iELAS	128	1242 × 375	57.5	3,428
[12]	R^3SGM + RES-BNN	128	1280 × 960	52	8,336
[14]	BNN + SGM	128	1280 × 960	64.15	10,089

Subsequently, we conducted a detailed comparative analysis of real-time performance across multiple FPGA platforms. Real-time capability is typically evaluated using million disparity estimates per second (MDE/s), a metric determined by frame rate (fps), image resolution, and disparity range. In our experiments, two resolutions (1242×375 and 1280×960) were tested with a disparity range of 128. Operating at 200 MHz, our system achieved throughputs of 3,185 MDE/s and 4,443 MDE/s respectively.

Compared with methods in [29, 38], our design demonstrates superior speed while maintaining competitive accuracy. When benchmarked against [35, 48], the system achieves comparable throughput but with significant improvements in matching precision, alongside support for higher resolutions and disparity ranges. In contrast to approaches in [40, 49], although our 8-bit quantized CNN-based implementation exhibits slightly lower throughput, it substantially enhances accuracy—particularly for images with rich details and complex textures—by capturing subtle structural features more effectively. Furthermore, the model retains numerical fidelity while enabling efficient FPGA-based inference, achieving an optimal balance between precision and real-time performance.

V. Conclusion

This paper presents an accelerator design based on 2D convolution cost features. To improve hardware efficiency in the CNN module, we adopted two key pipelining techniques: a systolic array architecture and a pipelined design for the CNN. We conducted a comprehensive evaluation of our local stereo matching system on the KITTI2015 stereo dataset, comparing it with other methods. The results indicated that our local method achieved an optimal balance among matching accuracy, hardware efficiency, and real-time performance.

REFERENCES

[1] Fan R, Jiao J, Pan J, et al. Real-time dense stereo embedded in a uav for road inspection[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2019: 0-0.

[2] Kamasaka R, Shibata Y, Oguri K. An FPGA-oriented graph cut algorithm for accelerating stereo vision[C]//2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE, 2018: 1-6.

[3] Hamid M S, Abd Manap N F, Hamzah R A, et al. Stereo matching algorithm based on deep learning: A survey[J]. Journal of King Saud University-Computer and Information Sciences, 2022, 34(5): 1663-1673.

[4] Mayer N, Ilg E, Hausser P, et al. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 4040-4048.

[5] Cao Y, Xu J, Lin S, et al. Gcnet: Non-local networks meet squeeze-excitation networks and beyond[C]//Proceedings of the IEEE/CVF international conference on computer vision workshops. 2019: 0-0.

[6] Cambuim L F S, Oliveira Jr L A, Barros E N S, et al. An FPGA-based real-time occlusion robust stereo vision system using semi-global matching[J]. Journal of Real-Time Image Processing, 2020, 17(5): 1447-1468.

[7] Liang Y, Lin D, Chen Z, et al. Research and implementation of adaptive stereo matching algorithm based on ZYNQ[J]. Journal of Real-Time Image Processing, 2024, 21(2): 46.

[8] Rahnama O, Cavalleri T, Golodetz S, et al. R3SGM: Real-time raster-respecting semi-global matching for power-constrained systems[C]//2018 International Conference on Field-Programmable Technology (FPT). IEEE, 2018: 102-109.

[9] Menze M, Geiger A. Object scene flow for autonomous vehicles[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3061-3070.

[10] Gao T, Wan Z, Zhang Y, et al. IELAS: An ELAS-based energy-efficient accelerator for real-time stereo matching on FPGA platform[C]//2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE, 2021: 1-4.

[11] Zhao J, Liang T, Feng L, et al. FP-Stereo: Hardware-efficient stereo vision for embedded applications[C]//2020 30th International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2020: 269-276.

[12] Ling Y, He T, Zhang Y, et al. Lite-stereo: a resource-efficient hardware accelerator for real-time high-quality stereo estimation using binary neural network[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(12): 5357-5366.

[13] Lu Z, Wang J, Li Z, et al. A resource-efficient pipelined architecture for real-time semi-global stereo matching[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021, 32(2): 660-673.

[14] Chen G, Ling Y, He T, et al. StereoEngine: An FPGA-based accelerator for real-time high-quality stereo estimation with binary neural network[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(11): 4179-4190.

[15] Lee Y, Choi S B, Lee E, et al. A real-time AD-census stereo matching based on FPGA[C]//2019 19th International Conference on Control, Automation and Systems (ICCAS). IEEE, 2019: 1622-1624.

[16] Firmansyah I, Yamaguchi Y. Fpga-based implementation of the stereo matching algorithm using high-level synthesis[C]//2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). IEEE, 2021: 1-7.